Riley Rutan
December 15th, 2024
GEOG 582 GIS Programming with Python
Final Project Report

4DVD Database Pipeline Python Package

1) Background

Climate change affects more people every year and poses a significant threat to humanity. While widely recognized and discussed, few people have worked with the underlying climate change data due to the technical barriers preventing people from accessing and analyzing it. The 4-Dimensional Visual Delivery of Big Climate Data (4DVD) is a web application built by the Climate Informatics Laboratory at SDSU to bridge this gap and provide user-friendly access to global climate data. An image of 4DVD displaying global air temperature in January 1851 is displayed in Figure 1.

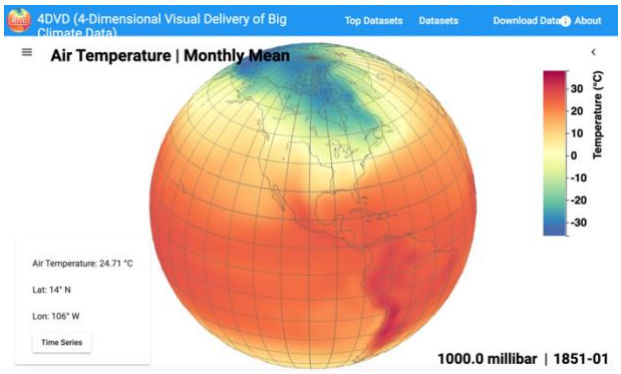

Figure 1: 4DVD Homepage

This application makes several historical and global climate datasets available to anyone, including sea surface temperature, precipitation, and air temperature anomalies, just to name a few. These datasets are stored in relational databases on a remote server at SDSU for reliable and blazing-fast data delivery. Many tools are included in 4DVD to analyze and visualize data, including time series analysis, statistical and climatological metrics, and linear modeling. The figure below displays a graphical interface after logging onto the 4DVD's server MySQL instance. On the left, each database corresponds to a dataset that is available for visualization in the application. The dataset 'Extended Reconstructed Sea Surface Temperature 5V Monthly Mean' database tables are shown on the right, with values of latitude, longitude, and date given unique ID values. The unique combination of lat and lon IDs creates a unique location, or GridBox_ID, and unique combinations of Gridbox_ID and Date_ID have a value of sea surface temperature. In other words, data that originates in the NetCDF format from NOAA is stored in the 4DVD database in a much different structure.

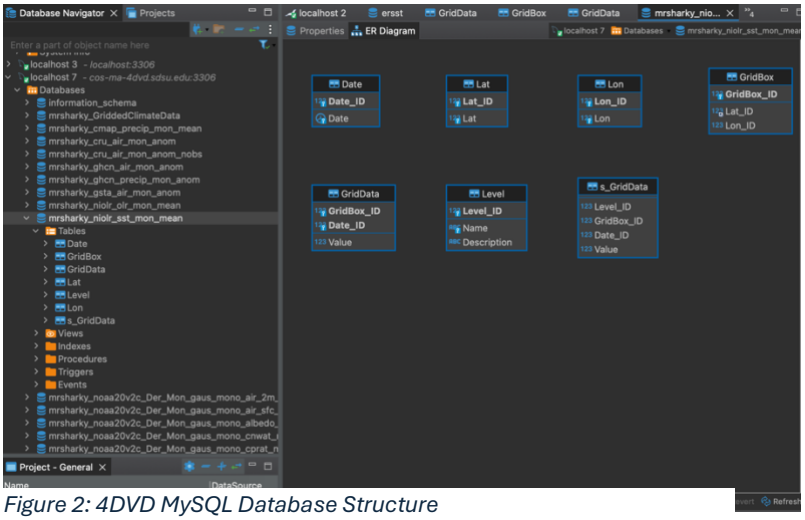

Figure 2: 4DVD MySQL Database Structure

2) Problem Statement

Recently, it was discovered that data is missing from the ERSST v5 dataset in both the 4DVD frontend and the database. This missing data is apparent in the white globe in Figure 3 when the date January 1912 is selected, as well as the Date_ID value jumping from 499 to 868 in the GridDate table in the database in Figure 4.

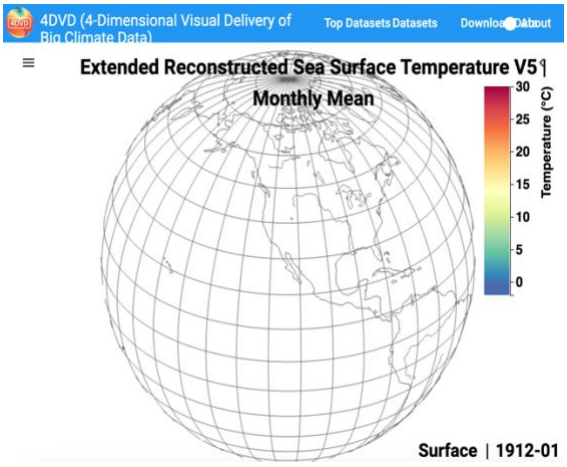

Figure 3: ERSST Missing Data

	GridBox_ID	Date_ID	Value
3676	3,676	499	8.05706
3677	3,677	499	7.453
3678	3,678	499	7.26866
3679	3,679	499	7.57726
3680	7,913	868	28.99126
3681	7,914	868	29.12447
3682	7,915	868	29.1982

Figure 4: ERSST Missing Data from Database

While the developers of 4DVD wrote code designed to transform and insert data into the database, it is no longer working as intended. That code is poorly documented, outdated, and broken, and the people who created it, used it, or understood it are no longer reachable. For this reason, a new pipeline is required to retrieve climate data from cloud storage, transform it to a format that matches the database architecture, and to connect to the database to insert new data. This new pipeline must include functions that are reusable, readable, and user-friendly, requiring only basic knowledge of Python rather than familiarity with the numerous languages and technologies that run 4DVD.

3) Code

3.1 get_data.py

This file serves as a first step in the pipeline and mostly uses the xarray python package. It allows the user to download NetCDF climate data directly from a URL endpoint and load it into Python. If the user already has this raster data on their local machine, the functions in this file can also load that data and extracts the specific variables that are required for the 4DVD database. These functions streamline downloading and processing large climate datasets and handle errors for missing files, invalid inputs, or failed downloads.

3.2 convert_data.py

This file contains functions that are focused on processing the data from a raster format into Pandas dataframes then to a format that matches the 4DVD database schema of relational tables. It loads the tables that are output from the database that are available locally or downloaded from the MySQL connection in the db_connect.py file. It then uses a series of transformations and merges to adjust longitude values, and match lat/lon/date with their corresponding unique IDs to reshape the data. The output is saved to a csv file that can be inserted in the GridData table in the database.

3.3 db_connect.py

This file contains all of the functions that interact with the MySQL database, and utilizes the mysql.connector python package. It allows the user to establish a connection to the database with their MySQL credentials. It allows users to export tables from a database to local csv files so data can be processed to match the database structure. It also allows the user to insert large datasets into tables very quickly with the LOAD DATA LOCAL INFILE command. This method has proven to achieve the fastest insert speeds when compared to other methods (Golotiuk, 2022).

3.4 test_functions.py and other files

The package contains a 'data' directory to organize all geospatial climate data that is used as an input or output for the package. There are three subdirectories named processed, raw, and 'db_structure' to organize the data further. There is a directory titled 'docs' to store all documents associated with this project. The License.txt file is created to keep my code free and open source, as is the spirit of Python. The README.md file was created with the assistance of ChatGPT by OpenAI to provide a summary of my package along with examples of its use for the homepage of a future GitHub site. The file requirements.txt manages the package's dependencies and lets any user easily install the required packages to avoid future conflicts. The file setup.py is used for packaging and distributing the project. It contains metadata and make it easy for people to install and use my code in the future. The __init__.py file marks the directory as a python package and makes all of the functions available for easier imports and usage of functions.

Finally, the test_functions.py file loads the package and tests all the functions. Different lines of this code are commented out, but they use all functions in the package to load data, transform it, and insert it into a duplicate 4DVD Database that was created on my localhost from a sql dump file. This was done to avoid creating any problems on the production 4DVD server while running tests.

Figure 5 displays the terminal output after the test_function.py file has run all functions to download raster climate data from the web, connected to the database and retrieve information about its structure, transform the raster data into tabular data to match database conventions, and insert this missing data back into the database. Figure 6 below shows the data that was missing is now present in the database GUI for the GridData Table. Date_ID values of 551 correspond to the date 1899-11.

```
rileyrutan@Rileys-Air Final_Project % /usr/local/bin/python3 /Users/rileyrutan/Desktop/SDSU/Semester_3/GIS_Python/Final_Project/tes
t_functions.py
Testing dload of netcdf data from NOAA from get_data.py
Testing loading local data, no download
Testing gettting db structure from local files
Testing getting db structure from mysql connection
Connected to MySQL database
Data from /Users/rileyrutan/Desktop/SDSU/Semester_3/GIS_Python/Final_Project/data/processed/gridded_data.csv inserted into GridData in 79.61 seconds
```

Figure 5: Pipeline Terminal Output

	GridBox_ID	Date_ID	Value
2321	2,321	551	6.80734
2322	2,322	551	6.80304
2323	2,323	551	6.69419
2324	2,324	551	6.55399
2325	2,325	551	6.58169
2326	2,326	551	6.83243

Figure 6: Inserted Data in Database

Conclusion

The development of this Python packages offers a solution to retrieve, transform, and insert climate data into the 4DVD database. By addressing the flaws of the previous software that was designed to complete this task, this package ensures that the user can manage big climate data with minimal technical expertise. It was designed to be readable, flexible, and simple to bridge the gap between database operations and common Python workflows. Hopefully future researchers can use this package to focus on climate data insights instead of wrestling with technical challenges. In the future, this package can be improved to include SSH tunneling, which will be required to connect to the production 4DVD server, not a test replica that I have been using. I believe I will be able to achieve this with the sshtunnel python package.

References

Golotiuk, D. (2022, August 8). *Inserting large data volumes into MySQL*. Medium. https://medium.com/datadenys/inserting-large-data-volumes-into-mysql-854df547a168